# THE HONG KONG POLYTECHNIC UNIVERSITY
# HONG KONG COMMUNITY COLLEGE

---

| | | | |
|---|---|---|---|
| **Subject Title** | : Data Structures | **Subject Code** | : SEHH2239 |
| **Session** | : Semester Two, 2020/21 | **Examination Time** | : 10:00 – 13:00 |
| **Date** | : 9 May 2021 | **Submission Period** | : 10:00 – 13:15 |
| **Subject Examiner(s)** | : Dr Pat CHAN<br>Mr Johnny TSUI | | |

---

This question paper has a total of **NINE** pages (including this covering page).

---

**Instructions to Candidates:**

1. There are TWO sections in this paper.
   - Section A (60%) – Short Questions. Answer any THREE out of FOUR questions in this section in the answer sheet provided. Each question carries 20 marks. If you answer more than three questions, only the first three attempted questions will be marked. Indicate in your answer sheet clearly which three questions you are attempting.
   - Section B (40%) – Long Questions. Answer the question in this section in the answer sheet provided. This question carries 40 marks.
2. Show all your work clearly and neatly. Marks will be deducted for untidy work.
3. Reasonable steps should be shown.
4. All programming code must be written in Java programming language.
5. You need to convert the answer sheet to PDF file before submission.

**Important points to follow:**

1. Please type your answers of this Take-home Examination in Microsoft Word.

2. Please strictly follow the **Submission Instructions** posted on Moodle before submission of your answer sheet. In addition,

- Please make sure that you have submitted the correct and entire file for the subject concerned. Useful information of PDF file generation is available at http://it-training.cpce-polyu.edu.hk/mod/book/view.php?id=1221.
- Please make sure there is no missing page in your submission.
- The file MUST include (a) full student name, (b) student number, (c) subject code, (d) subject group, (e) page number and (f) total number of pages on EVERY answer sheet.
- A submission period of 15 minutes is allowed after the examination end time for you to upload your completed answer sheet to Moodle. This is the Submission Deadline, and your answer scripts must be uploaded to Moodle via submission link before this Submission Deadline.

3. Please make sure that each uploaded page of your answer sheet is clearly captured. Only **ONE** single file in **PDF** format with less than **20MB** will be accepted.

4. Late submission via Moodle is not allowed.

5. Only the last submission you made within the designated timeslot will be counted.

6. Declaration of Original Work:

   By submitting the answer sheet of this Take-home Examination to the subject lecturer through Moodle, you hereby declare that the work in the answer sheet is completely your own work. No part of the answer sheet is taken from other people's work without giving them credit. All references have been clearly cited.

   You understand that an infringement of this declaration leaves you subject to disciplinary actions such as mark deduction, disqualification or even expulsion by the College.

   If necessary, students may be invited to provide more information on their submission.

**Section A (60%) – Short Questions**

**Answer any <u>THREE</u> out of FOUR questions in this section in the answer sheet provided. Each question carries 20 marks. If you answer more than three questions, only the first three attempted questions will be marked. Indicate in your answer sheet clearly which three questions you are attempting.**

<u>Question A1</u>

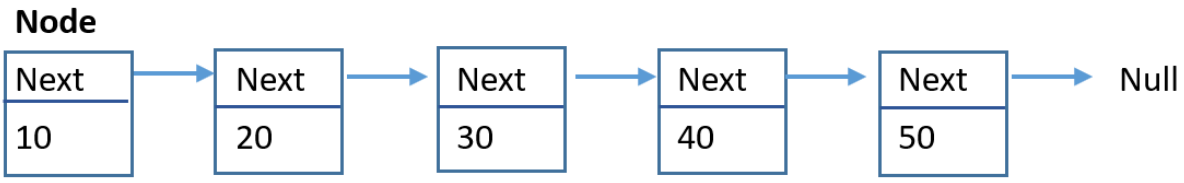



Figure 1: Linked list

Given the following class:

```
class Node {
    private Object element;    //data value
    public Node next;    //pointer to the next element of list,
                         // or null if last

   void RemoveNode (Node s, int n) {
   // remove number (n) of elements from the head of the list s
   :
   :
   }

   Node Attach(Node s1, Node s2) {
   // Attach the reverse of list s2 to the tail of the list s1
   :
   :
   }
}
```

(a) Complete the class ***Node*** with two constructors to initialize a ***Node*** for the following two cases: (i) only the *element* is given, (ii) both *element* and *next* (pointer to the next element) are given. (4 marks)

(b) Complete the method *RemoveNode* to remove *n* number of elements from the head of the list *s*. Make sure nothing will be returned or done by the method if the given *n* is less than zero or larger than the size of the list. (5 marks)

Question A1 (continued)

(c) Complete the method *Attach* by attaching the **reverse** of list *s2* to the **tail** of the list *s1* and return it. Make sure both lists *s1* and *s2* are not null, and the final concatenated list will be returned by the method. (11 marks)

Question A2

You are given a nearly sorted sequence 10, 1, 2, 3, 4, 5, 6, 7, 8, 9 to sort with the following sorting algorithm:

```
public static void sorting(Comparable [] a) {
    for (int i = a.length - 1; i > 0; i--)
                for (int j = 0; j < n; j++)
                if (a[j].compareTo(a[j+1]) > 0) {
                      Comparable temp = a[j];
                      a[j] = a[j+1];
                      a[j+1] = temp;
        }
}
```

(a) Identify the sorting algorithm. (2 marks)

(b) How many times of swapping is required at least for sorting the given number sequence with the sorting algorithm given? (4 marks)

(c) Modify the sorting algorithm given to reduce the number of swapping as minimum as possible. (8 marks)

(d) How many times of swapping is required for sorting the given number sequence with the modified sorting algorithm in (c)? (2 marks)

(e) Is the sorting algorithm able to sort the characters in this word "EXAMPLE"? Explain how the algorithm should be modified if necessary. (4 marks)

Question A3

Given the following three classes:

```
package ds;
class s1 {
    public int ex;

    s1(int p) {
      ex = 0;
    }

    Integer find(Integer[] numbers) {
     int ex = 0;
     Integer a = numbers[0];
     for (int i = 1; i < numbers.length-1; i++)
       if (a.compareTo(numbers[i+1]) > 0) {
         a = numbers[i];
         ++ex;
     }
     return a;
    }
}

package ds;
class s2 extends s1 {
    public int ex;

    s2(int p) {
      ex = 0;
    }

    Integer find(Integer[] numbers) {
      int ex = 0;
      Integer a = numbers[0];
      for (int i = 1; i < numbers.length-1; i++)
       if (a.compareTo(numbers[i+1]) < 0) {
          a = numbers[i];
          ++ex;
      }
      return a;
    }

    Integer finding(Integer[] numbers) {
        return super.find(numbers);
    }
}
```

Question A3 (continued)

```
package ds;
class tester {
  public static void main(String[] args) {
    int [] numbers = {1, 9, 5, 6, 3, 7, 4, 2};
    s2 obj = new s2(0);
    System.out.println("Ex: " + ex);
    System.out.println("Result: " + obj.finding(numbers));
  }
}
```

(a) State the relationship among these three classes. (3 marks)

(b) Highlight the mistakes, correct them, and state and explain the issues. (12 marks)

(c) Explain the function of the method *find* in ***S1*** and ***S2*** classes. (2 marks)

(d) What is the use of the variable *ex*? It is found that the output of the program is incorrect after successful execution. Why? How to fix it? (3 marks)

Question A4

(a) What is Hashing? (3 marks)

(b) Explain **ONE** popular hash function. (3 marks)

(c) Discuss the advantages and disadvantages of any **TWO** ways for solving collision in hashing. (6 marks)

(d) Draw the hash table with size 13 resulting from hashing the following keys:

25 48 71 12 95 22 25 18 89 10 56 81 65

Using the following hash function to solve the collision by **linked list**:

h(i) = (4i+7) mod 13

(8 marks)

- End of Section A -

**Section B (40%) – Long Questions**

**Answer the question in this section in the answer sheet provided. This question carries 40 marks.**

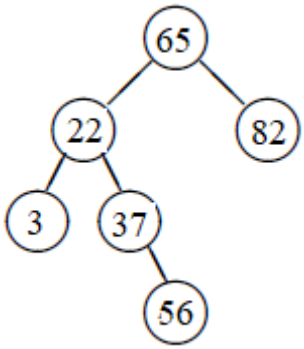
Question B1

Consider the following list of words:

```
amelia, tom, carmen, danny, yanny, felix, grace, harry
```

(a) Consider an initially empty binary search tree (BST). Place each of the above words in the order given above into the BST. (Use alphabetical order to make your comparisons.) Draw the completed binary search tree. (6 marks)

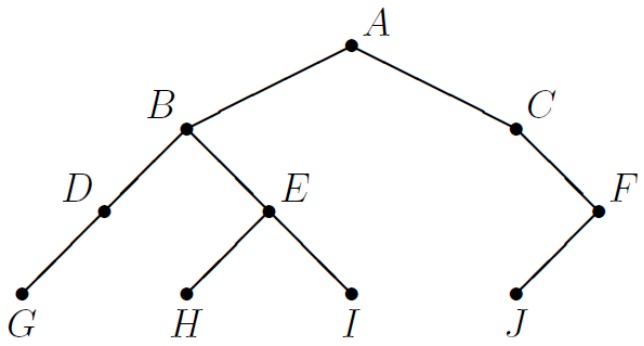
(b) Given the following tree that was obtained by inserting the element 56 into an AVL tree. The tree no longer satisfies the AVL invariant, but the invariant can be re-established by performing two rotate operations. Show the result after each rotation.




(6 marks)

Question B1 (continued)

(c) Given the following binary tree:




*A*-*J* should be formed with your student ID. The numbers A-*J* are generated as follows:

| Number | Based on your student ID number, formed by its … |
|---|---|
| *A* | 1st and 2nd digits |
| *B* | 2nd and 3rd digits |
| *C* | 3rd and 4th digits |
| *D* | 4th and 5th digits |
| *E* | 5th and 6th digits |
| *F* | 6th and 7th digits |
| *G* | 7th and 8th digits |
| *H* | 8th and 1st digits |
| *I* | 2nd and 4th digits |
| *J* | 6th and 8th digits |

(i) Draw the tree with the integers filled. (3 marks)

(ii) List the sequence of nodes visited by preorder traversal. (3 marks)

(iii) List the sequence of nodes visited by inorder traversal. (3 marks)

(iv) List the sequence of nodes visited by postorder traversal. (3 marks)

Question B1 (continued)

(d) You are given the following Java classes:

```
public class BTNode {
   int key;
   BTNode left; // left subtree
   BTNode right; // right subtree
   public BTNode(int key) {
       this.key = key;
   }
}

public class BTNodeTree {
   public BTNode r;
   public void postOrder(BTNode t) {
          // to be implemented
          System.out.print(t.key + " ");
   }
   public void insert(int theKey, BTNode t) {
          // to be implemented
   }
}
```

The `postOrder(BTNode t)` method displays every node it visits in a postorder traversal of the BST rooted at `t`. The `levelOrder(BTNode t)` method displays every node it visits in a level order traversal of the BST rooted at `t`. The `insert(int theKey, BTNode t)` method inserts `theKey` into the BST rooted at `t`.

(i) Complete the *postOrder* method according to the specification in this question.
(6 marks)

(ii) Complete the *insert* method according to the specification in this question.
(10 marks)

- End of Section B -

**- END OF PAPER -**